

# 深入浅出Flink (7)

---

## 一、课前准备

---

掌握上节课内容

## 二、课堂主题

---

本次课主要对Flink知识进行扩展

## 三、课程目标

---

1. Flink内存模型
2. SQL on Stream平台构建
3. Flink常见的面试题

# 21:25继续，大家休息13分钟

---

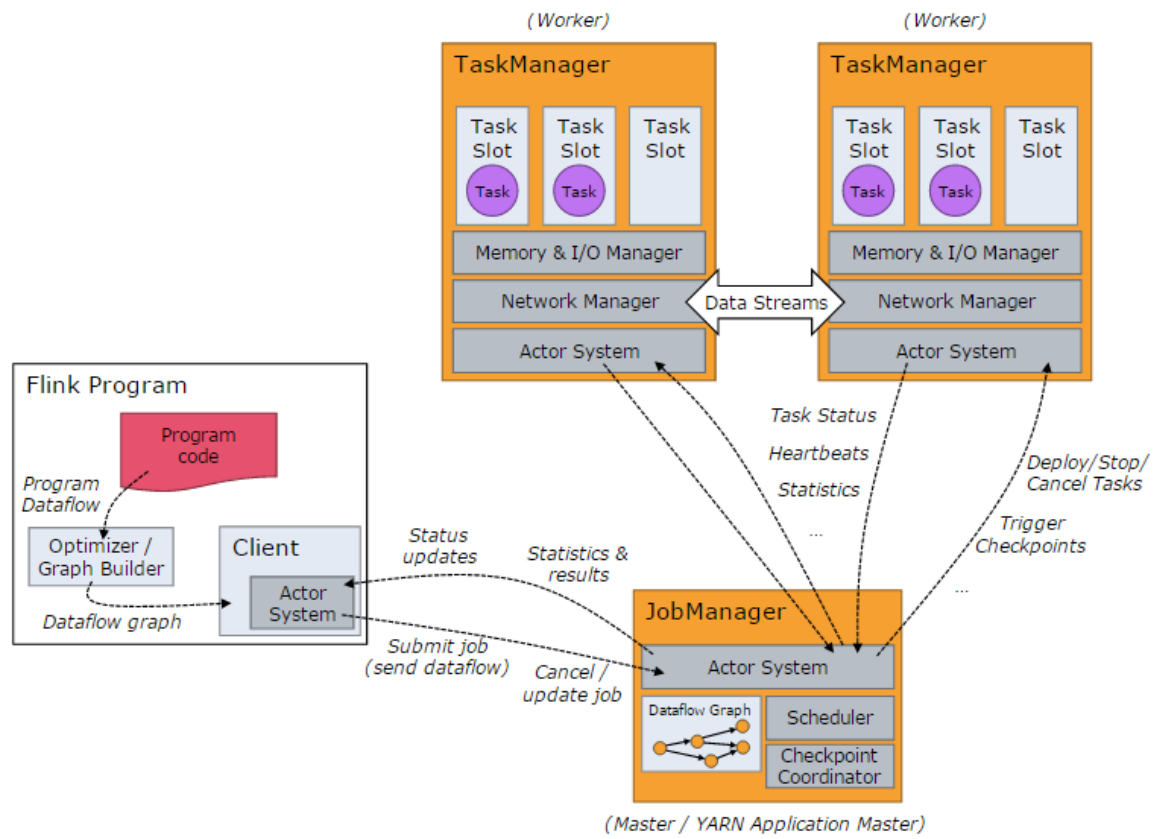
## 四、知识要点

---

### 4.1 Flink内存模型

#### 4.1.1 Flink运行流程回顾

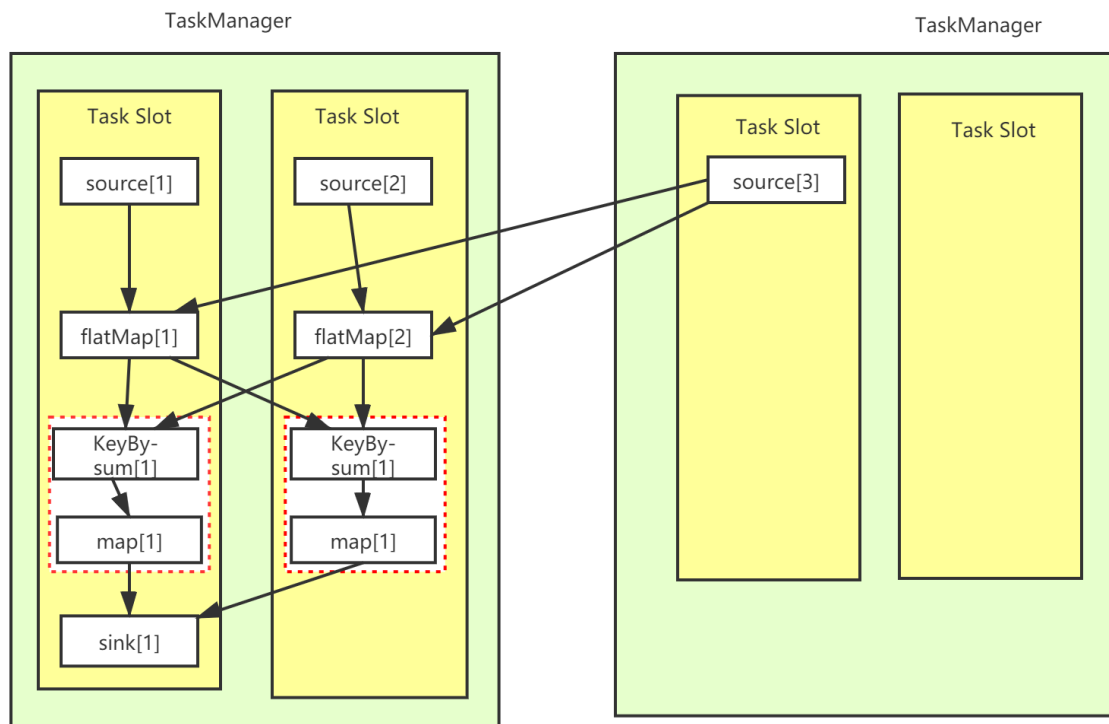
Flink任务运行流程



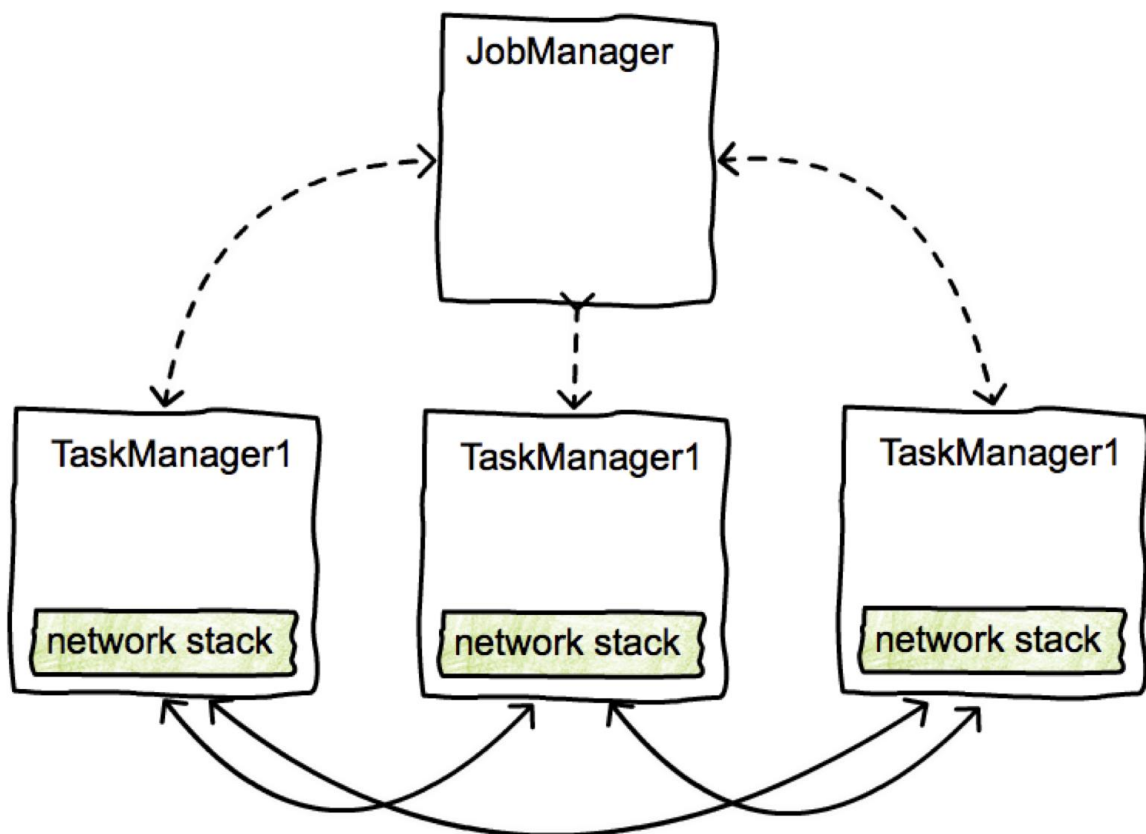
## DataFlow

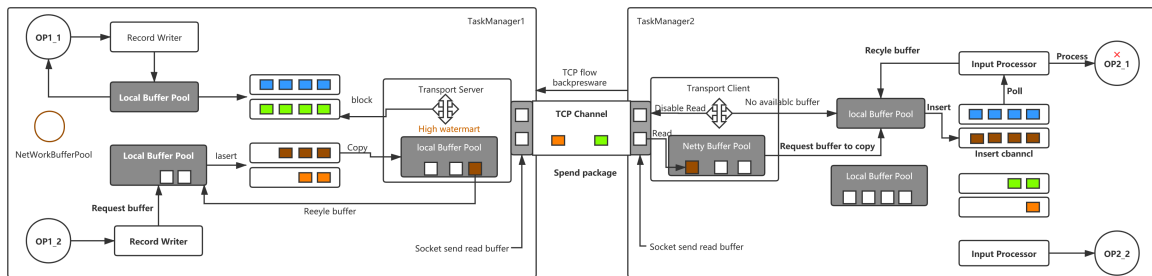


## Task



#### 4.1.2 TaskManager之间的数据传输



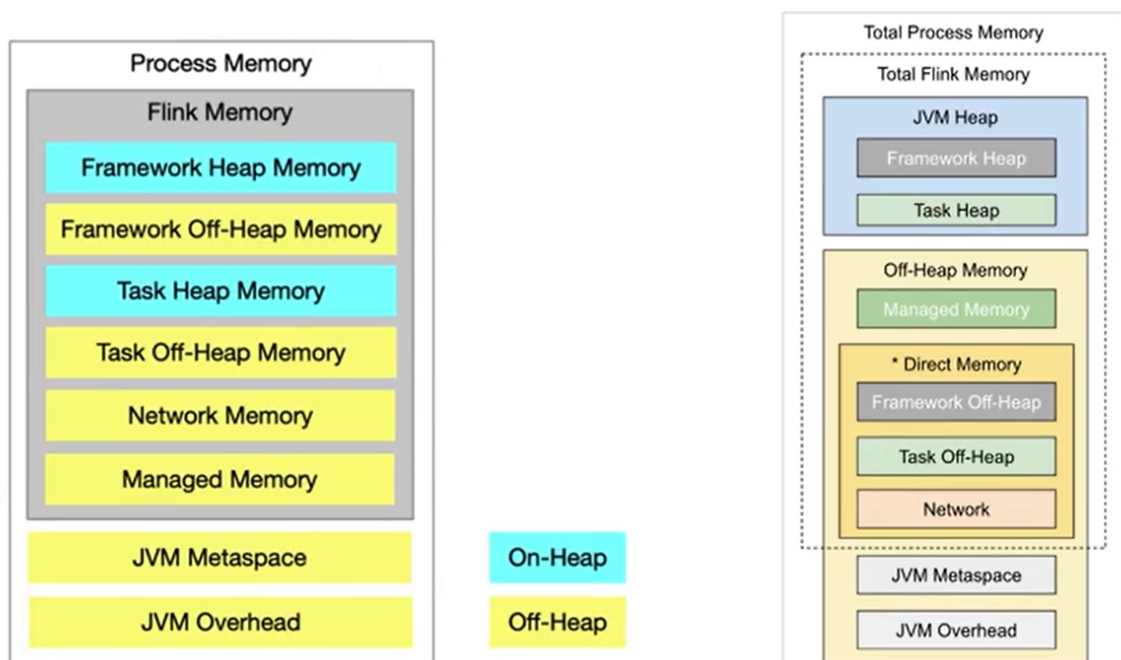


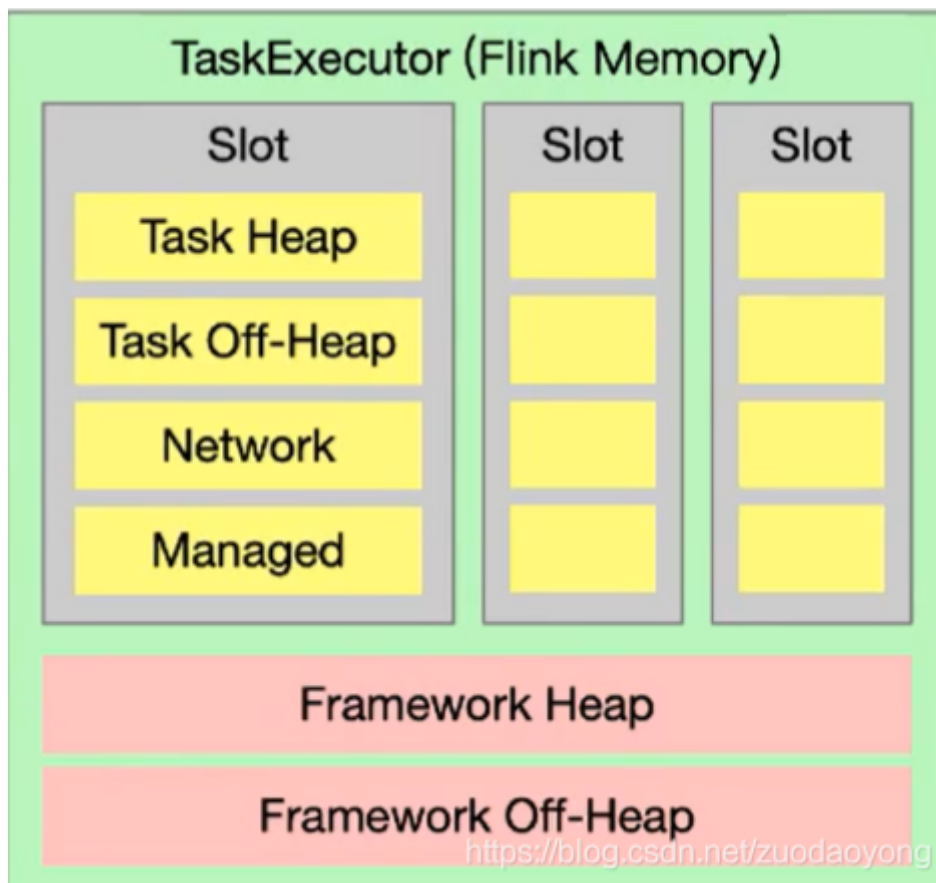
### 4.1.3 Flink内存管理

#### JVM管理内存的缺陷

1. Java对象存储密度低：一个只包含boolean属性的对象占用16个字节内存：对象头占了8个，boolean属性占了1个，对其填充占了7个，而实际只需要一个bit（1/8字节）就够了。
2. Full GC会极大地影响性能，尤其是为了处理更大数据而开了很大内存空间的JVM来说，GC会达到秒级甚至分钟级。
3. OOM问题影响稳定性：OutOfMemoryError是分布式计算框架经常会遇到的问题，当JVM中所有对象大小超过分配给JVM的内存大小时，就会发生OutOfMemoryError错误，导致JVM崩溃，分布式框架的健壮性和性能都会受到影响。

#### Flink内存模型





## 参数配置

组成	配置项	描述
Framework Heap Memory	taskmanager.memory.framework.heap.size	JVM堆内存专用于Flink框架(高级选项)
Task Heap Memory	taskmanager.memory.task.heap.size	JVM堆内存专用于Flink应用程序, 以运行算子和用户代码
Managed memory	taskmanager.memory.managed.size taskmanager.memory.managed.fraction	由Flink管理的native内存, 保留用于排序、哈希表、缓存中间结果和Rocks DB状态后端
Framework Off-heap Memory	taskmanager.memory.framework.off-heap.size	用于Flink框架的堆外direct(native)内存(高级选项)
Task Off-heap Memory	taskmanager.memory.task.off-heap.size	用于运行算子的Flink应用程序的堆外direct(native)内存
Network Memory	taskmanager.memory.network.min taskmanager.memory.network.max taskmanager.memory.network.fraction	为task之间的数据交换保留的直接内存(例如, 通过网络进行传输的缓冲)
JVM metaspace	taskmanager.memory.jvm-metaspace.size	Flink JVM进程的元空间大小
JVM Overhead	taskmanager.memory.jvm-overhead.min taskmanager.memory.jvm-overhead.max taskmanager.memory.jvm-overhead.fraction	为其他JVM开销保留的native内存:例如线程堆栈、代码缓存、垃圾收集空间等

## 4.2 SQL on Stream 平台介绍

### 4.2.1 背景

公司已经有iQuery平台, iQuery可以通过在页面写SQL语句, 通过Hive引擎, Spark引擎执行离线任务。故公司提出是否可以研发SQL on Stream平台, 实时任务也SQL化, 有如下好处:

1. 提升开发效率
2. 降低任务交接的难度
3. 降低实时任务开发门槛

于是研发了ZStream平台, 该平台基于<https://github.com/DTStack/flinkStreamSQL> 项目二次开发

```
sh submit.sh
-mode yarnPer
-sql /home/wen/Desktop/flink_stream_sql_conf/sql/Test01.sql
-name TestAll
-localSqlPluginPath /home/wen/IdeaProjects/flinkStreamSQL/plugins
-remoteSqlPluginPath /home/wen/IdeaProjects/flinkStreamSQL/plugins
-flinkconf /home/wen/Desktop/flink_stream_sql_conf/flinkConf
-yarnconf /home/wen/Desktop/flink_stream_sql_conf/yarnConf_node1
-flinkJarPath /home/wen/Desktop/dtstack/flink-1.8.1/lib
-pluginLoadMode shipfile
-confProp {"time.characteristic\":\"eventTime\",\"logLevel\":\"info\"}
-queue c
```

## ZStream页面

The screenshot shows the 'ZStream' management page. At the top is a navigation bar with links like '首页', '数据开发', '元数据', '监控', '工单', '数据查询', '系统设置', and '集群管理'. Below this is a sub-header with '操作说明书', '任务管理', and '新建任务'. The main content area has a search bar with filters for '代号' (set to '全部'), '创建人' (set to '李希沅(lixiyuan)'), and '名称'. Below the search bar is a table of tasks.

ID	名称	创建人	代号	监控	状态	YARN LOG	运行日志	操作	创建时间
44	测试socket任务	lixiyuan	zdp		KILL	运行日志		编辑 启动 删除 复制	2019-11-12 17:25
43	TabRec点击流预处理_升级测试	lixiyuan	zdp		运行时失败	运行日志		编辑 启动 删除 复制	2019-11-12 16:49
42	测试吕实时ETL_duplicated2_duplicated	lixiyuan	zdp		新建			编辑 启动 删除 复制	2019-09-20 14:48
40	用户最近活跃时间_duplicated2	lixiyuan	zdp		新建			编辑 启动 删除 复制	2019-09-02 10:49
39	用户最近活跃时间_duplicated	lixiyuan	zdp		新建			编辑 启动 删除 复制	2019-08-20 17:12
38	测试redis	lixiyuan	zdp		KILL	运行日志		编辑 启动 删除 复制	2019-08-19 11:53
37	testrowtime_duplicated	lixiyuan	zdp		KILL	运行日志		编辑 启动 删除 复制	2019-08-19 11:50

## 新建任务

The screenshot shows the '新建任务' (New Task) form. It has a navigation bar with '任务管理', '新建任务', and 'UDX管理'. Below the navigation bar are tabs for '基本信息', 'SQL', and '参数'. The '基本信息' tab is active. The form contains the following fields:

- 任务名称: 任务名
- 负责人: lixiyuan
- 接警人: 李希沅(lixiyuan)
- 任务描述: (Empty text area)

At the bottom, there are buttons for '返回任务管理' and '保存'.

## SQL编辑

操作说明书

任务管理新建任务UDX管理

基本信息SQL参数

校验SQL语法

提示：若SQL中使用了UDX函数，请提前引入对应的UDX函数，再校验。

使用SQL模板

1

引用UDX自定义函数文件

SQL中的自定义函数必须要引入对应的函数文件，若不存在请上传。

返回任务管理

## 参数填写

首页数据开发元数据监控工单数据查询系统设置集群管理李希沅

操作说明书任务管理新建任务UDX管理

基本信息SQL参数

集群：flink-1.6

TaskManager内存(M)：1024

slots/TaskManager：5

TimeCharacteristic：ProcessingTime

☒ 开启Checkpoint：Checkpoint间隔(s)：建议：30-120秒左右。Checkpoint模式：EXACTLY\_ONCE

队列：root.online.dp

TaskManager个数：2

并行度：10

重试：0

延迟报警

返回任务管理

保存

## UDF管理

首页数据开发元数据监控工单数据查询系统设置集群管理李希沅

操作说明书任务管理新建任务UDX管理

查询

代号：全部创建人：李希沅(lixiyuan)类型：全部

名称：名称

查询


列表

显示 10 项结果

ID	名称	类型	创建人	代号	描述信息	时间	操作
11	getMessage.jar	UDTF	lixiyuan	zdp	获取日志信息	2019-07-10 19:23:47	删除
10	flinkudf-1.0-SNAPSHOT.jar	UDF	lixiyuan	zdp	xx	2019-07-10 19:08:45	删除

显示第 1 至 2 项结果，共 2 项

上页1下页

 首页 数据开发 元数据 监控 工单 数据查询 系统设置 集群管理 李希沅

操作说明书 任务管理 新建任务 UDX管理

基本信息 SQL 参数 运行历史

校验SQL语法 提示：若SQL中使用到了UDX函数，请提前引入对应的UDX函数，再校验。 使用SQL模板

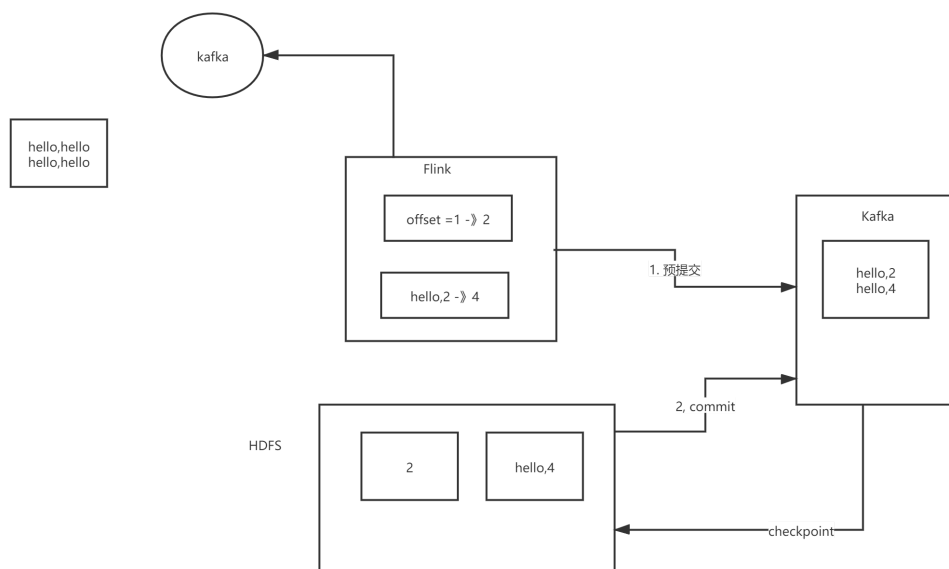
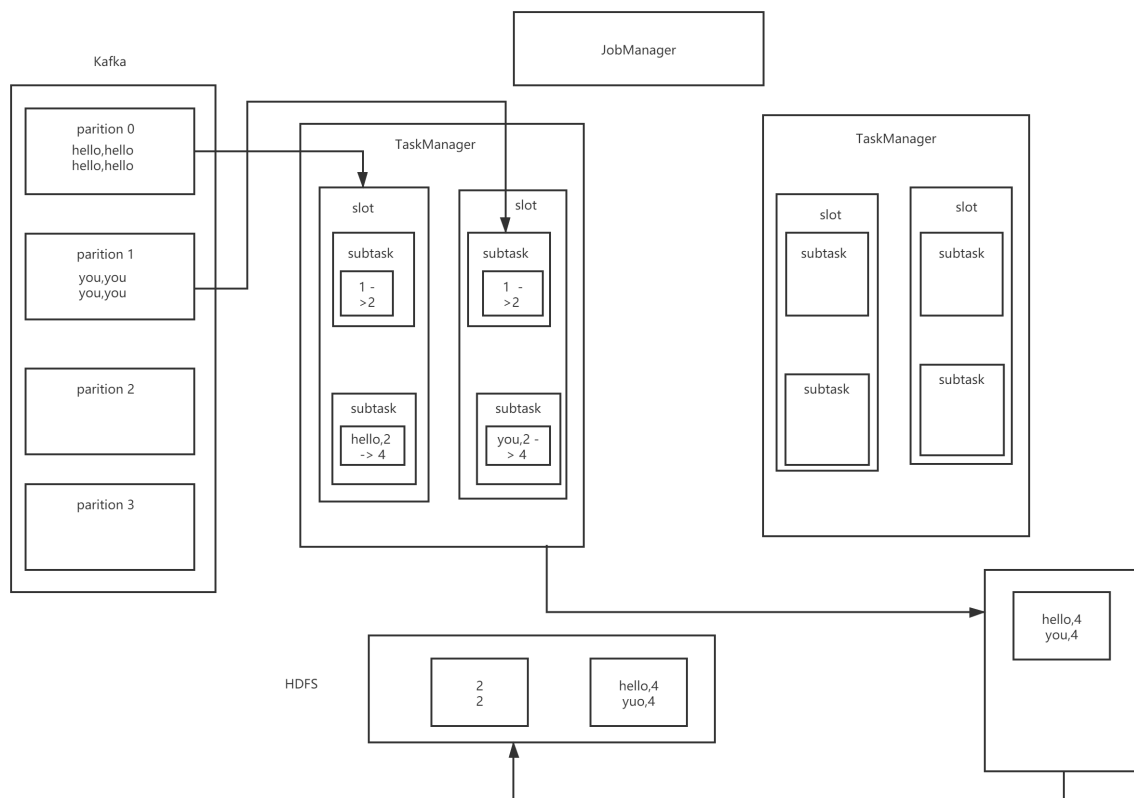
```
1 CREATE TABLE kafka_log(  
2   _topic varchar,  
3   _messageKey varchar,  
4   _message varchar,  
5   _partition INT,  
6   _offset BIGINT  
7 )WITH(  
8   type='kafka11',  
9   kafka.bootstrap.servers='10.1.16.15:2181,10.1.16.15:2182,10.1.16.15:2183',  
10  kafka.zookeeper.quorum='10.1.16.15:2181,10.1.16.15:2182,10.1.16.15:2183',  
11  kafka.auto.offset.reset='latest',  
12  kafka.topic='hdp_ubu_zhuanzhuan_infologic',  
13  parallelism='10',  
14  kafka.group.id='ttttt',  
15  sourcedatatype='text'  
16 );  
17  
18 CREATE TABLE MyResult(  
19   testtopic varchar
```

引用UDX自定义函数文件  
SQL中的自定义函数必须要引入对应的函数文件，若不存在请上传。

## 五、 常见面试题(出去面试了)

1. Flink从Kafka读取数据，把处理结果写入Redis/HBase/Kafka，如何保证Exactly-once？





## 2. 如何设计一个SQL on Stream的平台？ 概率比较小

这个问题，其实就是面试官，他想做，然后自己不知道怎么做，想通过你，启发一下他，需求一些经验。

用刚刚的内容回答就行，说思路就可以。

## 3. Flink相比SparkStreaming有什么区别

## 1. 架构模型

Spark Streaming 在运行时的主要角色包括：Master、Worker、Driver、Executor

Flink 在运行时主要包含：Jobmanager、Taskmanager、Client、Slot。

## 2. 任务调度

Spark Streaming 连续不断的生成微小的数据批次，构建有向无环图 DAG，Spark Streaming 会依次创建 DStreamGraph、JobGenerator、JobScheduler。

Flink 根据用户提交的代码生成 StreamGraph，经过优化生成 JobGraph，然后提交给 JobManager 进行处理，JobManager 会根据 JobGraph 生成 ExecutionGraph，ExecutionGraph 是 Flink 调度最核心的数据结构，JobManager 根据 ExecutionGraph 对 Job 进行调度。

## 3. 时间机制

Spark Streaming 支持的时间机制有限，只支持处理时间。

Flink 支持了流处理程序在时间上的三个定义：处理时间、事件时间、注入时间。同时也支持 watermark 机制来处理滞后数据。

## 4. 容错机制

对于 Spark Streaming 任务，我们可以设置 checkpoint，然后假如发生故障并重启，我们可以从上次 checkpoint 之处恢复，但是这个行为只能使得数据不丢失，可能会重复处理，不能做到恰一次处理语义，如果我们想要实现 Exactly-once，需要自己实现。

Flink 内部提供了 Exactly-once 实现，使用起来较为方便

## 5. 还可以有其他角度

state, window 等等

## 4. Flink 的三种语义是什么？

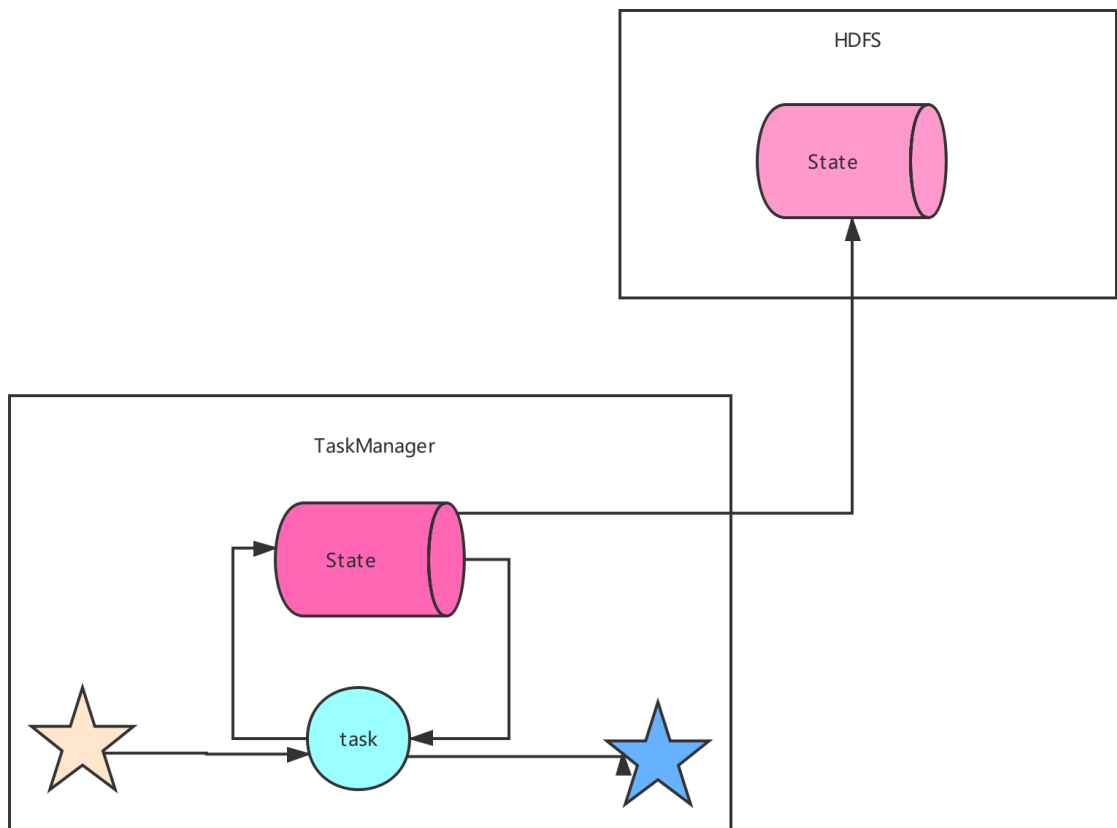
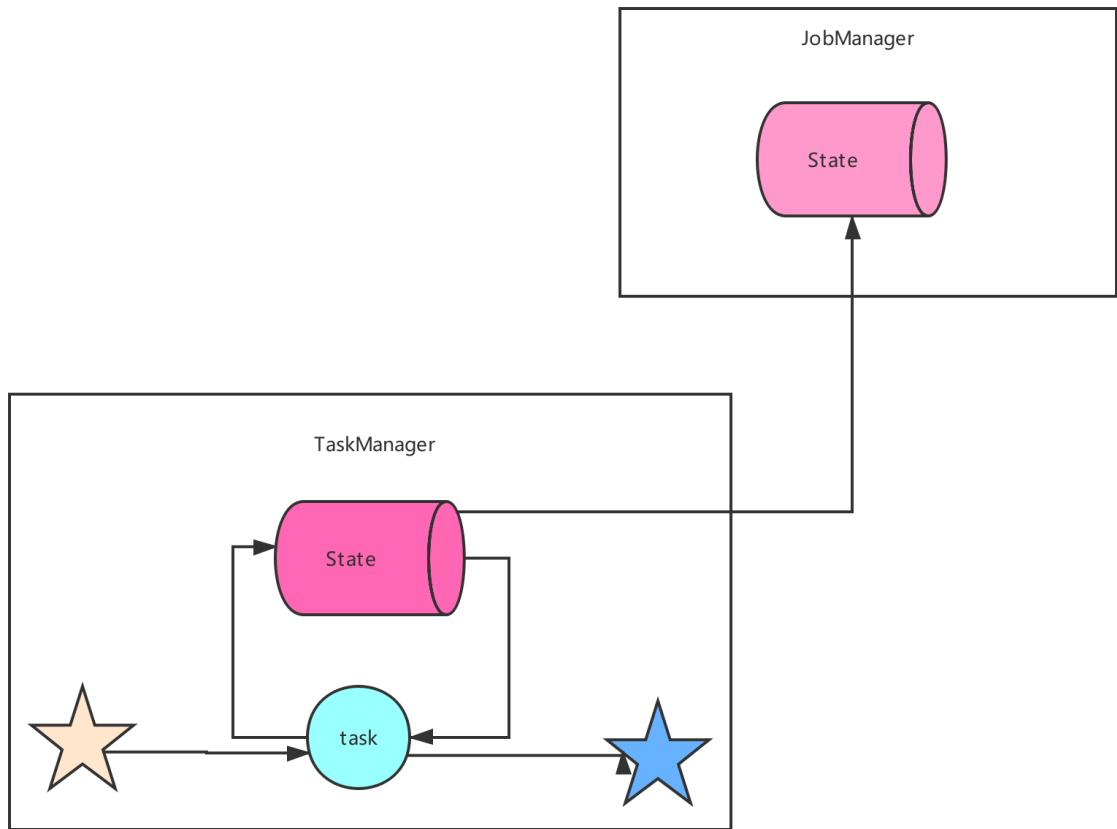
1. **Event Time**: 这是实际应用最常见的时间语义。
2. **Processing Time**: 没有事件时间的情况下，或者对实时性要求超高的情况下。
3. **Ingestion Time**: 存在多个 **Source Operator** 的情况下，每个 **Source Operator** 可以使用自己本地系统时钟指派 **Ingestion Time**。后续基于时间相关的各种操作，都会使用数据记录中的 **Ingestion Time**。

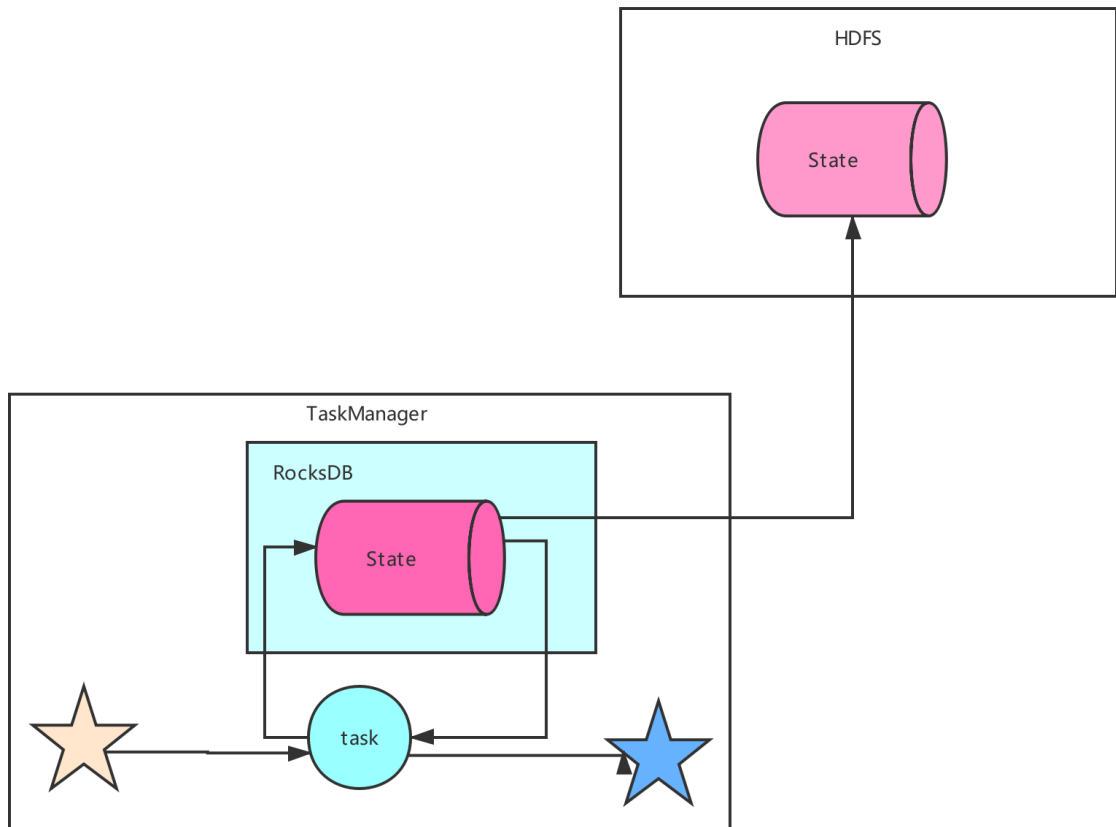
## 5. Flink 场景的重启策略有哪几种？

Flink 实现了多种重启策略，常见的有如下三种：  
固定延迟重启策略（**Fixed Delay Restart Strategy**）  
故障率重启策略（**Failure Rate Restart Strategy**）  
没有重启策略（**No Restart Strategy**）

## 6. 说说 Flink 的状态存储

Flink 提供了三种状态存储方式：**MemoryStateBackend**、**FsStateBackend**、**RocksDBStateBackend**。





### 7.Flink 中 exactly-once 语义是如何实现的，状态是如何存储的？

- 1) Flink内部 依靠 checkpoint 机制来实现 exactly-once 语义，
- 2) 如果要想实现端到端的 exactly-once，还需要外部 source 和 sink 满足一定的条件。状态的存储通过状态后端来管理，Flink 中可以配置不同的状态后端。

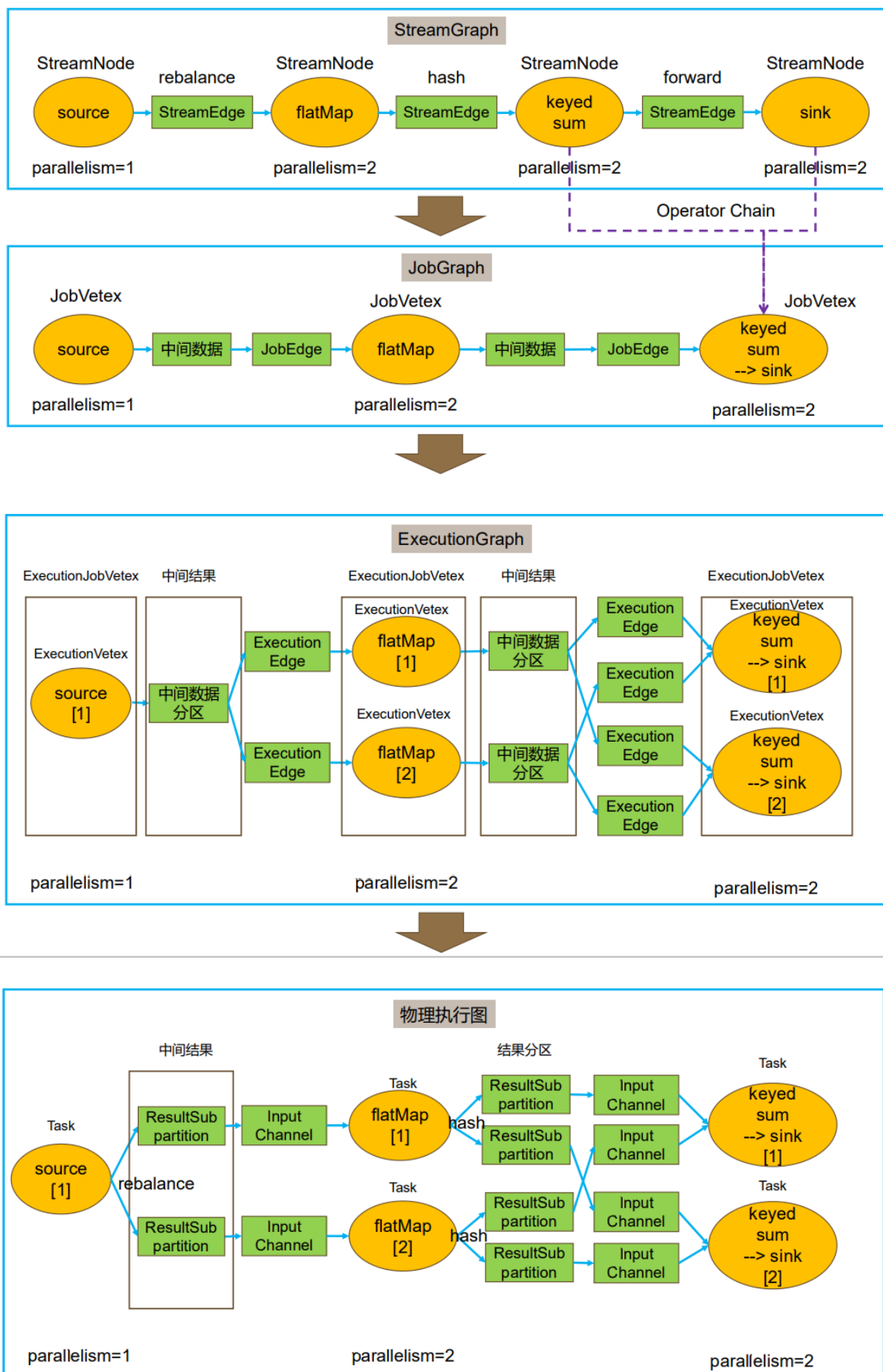
### 8.Flink 的 checkpoint 机制对比 spark 有什么不同和优势？

spark streaming 的 checkpoint 仅仅是针对 driver 的故障恢复做了数据和元数据的 checkpoint。而 flink 的 checkpoint 机制要复杂了很多，它采用的是轻量级的分布式快照，实现了每个算子的快照，及流动中的数据快照。

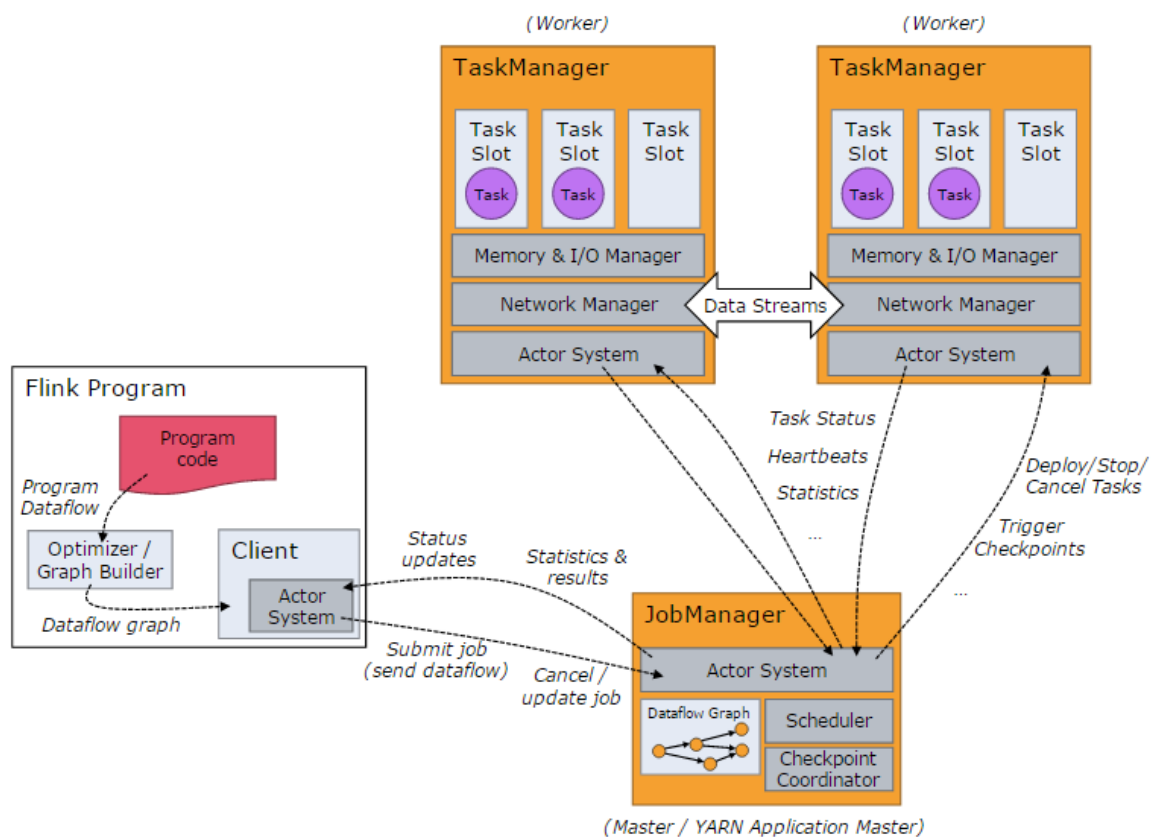
### 9. Flink怎么去重？考虑一个实时场景：双十一场景，滑动窗口长度为 1 小时，滑动距离为 10 秒钟，亿级用户，怎样计算 UV？

使用类似于 scala 的 set 数据结构或者 redis 的 set 显然是不行的，因为可能有上亿个 key，内存放不下。所以可以考虑使用布隆过滤器（Bloom Filter）来去重。

### 10. 说说flink的三层模型



## 11. 说说Flink任务的运行流程



## 12.如果Flink任务延迟高，你有什么优化的思路？

最主要的手段是资源调优和算子调优。资源调优即是对作业中的Operator的并发数（parallelism）、CPU（core）、堆内存（heap\_memory）等参数进行调优。作业参数调优包括：并行度的设置，State的设置，checkpoint的设置

## 六、作业